МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Автономна сонячна електростанція з системою позиціонування за сонцем»
здобувача групи ІН-06-2 Чечелюка Богдана Андрійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Богдан ЧЕЧЕЛЮК
(підпис)

Керівник,
старший викладач кафедри
комп'ютерних наук, к.ф.-м.н. Дмитро ВЕЛИКОДНИЙ _____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Чечелюк Б. А.

1. Тема роботи: «Автономна сонячна електростанція з системою позиціонування за сонцем» затверджую наказом по СумДУ від -_____

2. Термін здачі здобувачем кваліфікаційної роботи -_____

3. Вхідні дані до кваліфікаційної роботи -_____-

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження.
2) Огляд технологій, що використовуються для розбробки інформаційного та програмного забезпечення автономної сонячної станції з системою позиціонування за сонцем 3) Розробка автономної сонячної станції з системою позиціонування за сонцем 4) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, для розробки автономної сонячної електростанції з системою позиціонування за сонцем*		
3	*Розробка автономної сонячної електростанції з системою позиціонування за сонцем*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 65 стор., 29 рис., 2 додатка, 28 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі підвищення ефективності використання сонячної енергії шляхом розробки автономної сонячної електростанції з системою позиціонування за сонцем.

Об’єкт дослідження — Об’єктом дослідження є процес оптимізації виробництва електроенергії сонячними електростанціями шляхом впровадження системи стеження за сонцем.

Мета роботи — розробка автономної сонячної електростанції з системою позиціонування за сонцем, що дозволить максимізувати збір сонячної енергії протягом дня та підвищити загальну ефективність електростанції.

Методи дослідження — аналіз та моделювання траєкторії руху сонця, розробку та впровадження алгоритмів автоматичного стеження за сонцем, а також технічні засоби для реалізації системи позиціонування.

Результати — розроблено систему позиціонування, яка зчитує дані струму з вольтметрів та амперметрів, обробляє їх, надає змогу перегляду показників через локальну мережу або телеграм бота, зчитує показники фоторезисторів які дозволяють дізнатись місцезнаходження сонця відносно станції, має кнопки керування положенням панелі.

СИСТЕМА ПОЗИЦІОНУВАННЯ, СОНЯЧНИЙ ТРЕКЕР, ЕНЕРГЕТИКА, ARDUINO, TELEGRAM, OTA.

Зміст

ВСТУП **5**
1 ІНФОРМАЦІЙНИЙ ОГЛЯД **6**
1.1 Аналіз предметної області **6**
1.2 Види електростанцій **9**
1.3 Елементна база сонячних електростанцій **11**
1.4 Постановка задачі **18**
2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ **20**
2.1 Аналіз середовищ розробки **20**
2.2 Середовище розробки **22**
2.3 Інструменти створення ботів **26**
3 ПРОГРАМНА РЕАЛІЗАЦІЯ Й ТЕСТУВАННЯ **32**
3.1 ПЗ для розробки программ **32**
3.2 Розробка програмного забезпечення для плат **37**
3.3 Тестування системи **44**
Висновки **48**
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **49**
Додаток А **52**
Додаток Б **62**

ВСТУП

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи належить до актуальних напрямів розвитку інформаційних технологій, оскільки в сучасному світі велике значення набувають автономні сонячні електростанції. Вони є перспективним рішенням, оскільки забезпечують ефективне використання сонячної енергії і можуть бути встановлені в будь-яких регіонах, де є доступ до сонячної радіації. Така технологія відповідає сучасним екологічним та енергетичним вимогам, сприяючи зменшенню викидів шкідливих речовин та залежності від традиційних джерел енергії.

Об’єкт дослідження – процес автоматичного запуску та керування системою позиціонування за сонцем.

Мета роботи – розробка інструменту для керування системою позиціонування за сонцем. Ця система має на меті забезпечити зручний та ефективний процес моніторингу та керування додатками навіть за межами локальної мережі.

Методи дослідження – у роботі використовуються методи аналізу та розробки програмного забезпечення, зокрема використання інструментів моделювання та симуляції, експериментальних вимірювань, аналізу даних, досліджень інженерних параметрів та спостереження в реальних умовах.

Результати – розроблено програмне забезпечення, яке дозволяє автоматизувати процес запуску та управління системи спостереження за сонцем. Система включає в себе інструменти для ефективного моніторингу та керування, що дозволяє забезпечити максимальну ефективність використання сонячного випромінювання протягом дня.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

В сучасному світі у людства виникає дуже велика потреба в електроенергії. Кожен день збільшується кількість електроприладів що під час одночасного включення потребують багато енергії. Зазвичай її виробляють за допомогою атомної енергетики, також є вуглевипалювальні та інші станції для забезпечення потрібного об’єму. Але чим це загрожує людству? Кожен день в повітрі збільшується кількість відходів що генеруються через роботу таких станцій. Тож постає питання, а як можна забезпечити велику кількість енергії більш чистим способом?

Сонячні панелі є одним з таких способів. Вони використовують фотоелементи які перетворюють сонячну енергію в постійний струм що відрізняється від сонячних колекторів, котрі нагрівають матеріал-теплоносій. Якщо встановити рухому панель, вона буде більш ефективна за звичайну статичну, адже працює на сонячних трекерах, які мають кілька способів роботи:

1. Ручний - управління кутом нахилу та вибір кількості поворотів здійснюється оператором. Продуктивність СЕС залежить від його досвіду та кваліфікації.
2. Пасивний - орієнтація проводиться за заданим алгоритмом для конкретного географічного розташування фоторезисторів.
3. Активний – система орієнтує робочу поверхню модуля перпендикулярно до сонця.

Таким чином можна досягти максимальної ефективності панелей та берегти природу з вичерпними твердопаливними ресурсами. Сонячні панелі мають такі переваги [1]:

1. Відновлювана енергія: сонячна енергія є відновлювальним джерелом енергії, оскільки сонце постійно випромінює енергію без обмежень, надійно забезпечуючи потреби в електроенергії.

2. Екологічночиста енергія: виробництво електроенергії за допомогою сонячних панелей не створює шкідливі речовини або токсичні відходи, що сприяє зменшенню вуглецевого сліду та покращенню якості навколишнього середовища.
3. Незалежність від енергетичних мереж: сонячні панелі дозволяють створювати електроенергію на місці використання, що дозволяє знизити залежність від централізованих енергетичних мереж та забезпечити електрику там, де вона найбільше потрібна.
4. Низькі операційні витрати: після встановлення сонячних панелей витрати на виробництво електроенергії мінімальні, оскільки сонце, як джерело енергії, є безкоштовним.
5. Довговічність та низьке обслуговування: сонячні панелі мають довгий термін служби (зазвичай більше 25 років) і вимагають мінімального обслуговування, що знижує загальні витрати на утримання.
6. Гнучкість в установці: сонячні панелі можна встановлювати на дахах будівель, на відкритих майданчиках, на землі або в морі, що забезпечує гнучкість у їхньому розташуванні та використанні.
7. Збереження енергії: інтеграція систем зберігання енергії, таких як акумуляторні батареї, дозволяє зберігати електроенергію, що виробляється сонячними панелями, для використання в періоди поганої погоди або в нічний час.

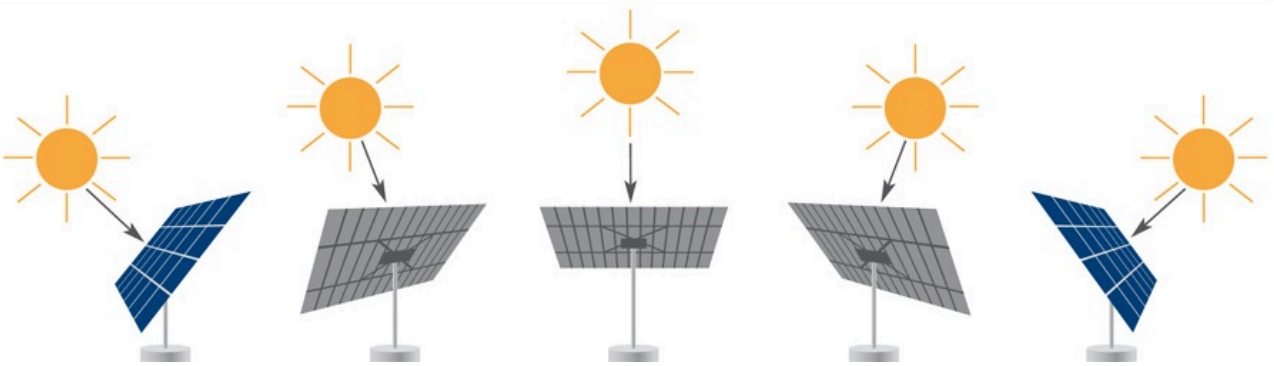

Рисунок 1.1 – Приклад роботи автоматизованої сонячної панелі

Сонячні панелі стали особливо популярними в енергетичній галузі завдяки їхній здатності забезпечувати електроенергією без викидів шкідливих речовин та інших негативних впливів на навколишнє середовище.

Вони можуть працювати в різних умовах, надаючи електроенергію навіть в умовах поганої погоди. Це робить їх ефективними та надійними джерелами енергії для використання у всіх сферах життя.

Забезпечують електроенергією різних споживачів, включаючи побутові, комерційні та промислові об'єкти. Їхнє використання сприяє зменшенню вартості електроенергії та зниженню викидів вуглецю в атмосферу.

Їх зазвичай встановлюють на будинках, громадських спорудах, сільськогосподарських об'єктах та інших місцях, де доступно сонячне випромінювання. Це робить їх універсальними та широкозастосовуваними в різних галузях енергетики.

Рисунок 1.2 – Приклад роботи сонячної панелі

1.2 Види електростанцій:

Сонячні електростанції можна класифікувати за різними критеріями, включаючи спосіб підключення до електричної мережі та незалежність від неї. Детальний опис обох типів [2]:

- **Мережеві сонячні електростанції:**

Мережеві сонячні електростанції, часто називають також "онлайн" або "підключені до мережі", працюють у взаємодії з електричною мережею. Основна їх функція полягає в генерації електроенергії, яка потім передається в електричну мережу для використання споживачами. Основні компоненти мережевих сонячних електростанцій включають:

- Сонячні панелі: Вони збирають сонячне випромінювання і перетворюють його на електричну енергію.
- Інвертори: Ці пристрої конвертують постійний струм, що генерується сонячними панелями, в змінний струм, який використовується в електричній мережі.
- Мережевий підключений зв'язок: Це забезпечує зв'язок між сонячною електростанцією та електричною мережею.
- Лічильник електроенергії: Він вимірює кількість електроенергії, яку генерує сонячна електростанція та яка передається в мережу.

Мережеві сонячні електростанції є ефективним способом виробництва електроенергії, оскільки вони можуть споживати мережеву електроенергію у тих випадках, коли сонце не сяє достатньо яскраво, а також надлишкову енергію можна продавати назад у мережу.

- **Автономні сонячні електростанції:**

Автономні сонячні електростанції, також відомі як "незалежні" або "офф-грід", працюють незалежно від електричної мережі. Вони зазвичай використовуються в регіонах, де немає доступу до централізованої електричної

мережі або де підключення до мережі є недоцільним з фінансових або інфраструктурних причин. Основні компоненти автономних сонячних електростанцій включають:

- Сонячні панелі: Вони збирають сонячне випромінювання і генерують електроенергію.
- Батареї: Використовуються для зберігання електроенергії, яка генерується сонячними панелями для використання в періоди, коли сонце не світить або вночі.
- Контролер заряду: Керує процесом заряду батарей і запобігає їх перезарядці або перерозрядці.
- Інвертор: Як і у мережевих сонячних електростанціях, інвертор використовується для конвертації постійного струму у змінний струм, щоб його можна було використовувати для побутових потреб.
- Резервне джерело енергії: Деякі системи можуть мати дизельні генератори або інші резервні джерела енергії, які використовуються в разі виникнення проблем із сонячними панелями або батареями.

Автономні сонячні електростанції є чудовим рішенням для вирішення енергетичних потреб у віддалених районах або в тих випадках, коли доступ до мережі обмежений або неможливий. Вони можуть бути ефективними в різних ситуаціях, від сільськогосподарських господарств до кемпінгів і побутових будинків.

1.3 Елементна база сонячних електростанцій:
Trina Solar Bifacial 645W фотомодуль електричний

Trina Solar Bifacial 645W- це інноваційна сонячна панель, розроблена з використанням передових технологій, щоб забезпечити найвищу ефективність та надійність. Ця панель вирізняється своєю двосторонньою конструкцією, яка дозволяє збирати сонячну енергію з обох сторін панелі, забезпечуючи її високу потужність та виробництво енергії [3].

Особливості:

1. Двостороння технологія: панель Trina Solar Bifacial 645W може збирати сонячну енергію з обох сторін завдяки своїй біфаціальній конструкції. Це дозволяє використовувати не тільки пряме сонячне випромінювання, але й відбите випромінювання, що проникає через задню сторону панелі. Це підвищує загальну потужність та енергетичний вихід панелі.
2. Висока потужність: завдяки своїй потужності 645 ватт, Trina Solar Bifacial 645W забезпечує високу енергетичну продуктивність. Це дозволяє забезпечити більше сонячної енергії на квадратний метр та знизити кількість необхідних панелей для досягнення потрібного рівня енергетичного виробництва.
3. Висока ефективність при низькому освітленні: Trina Solar Bifacial 645W має високу ефективність навіть при низькому рівні освітлення. Це означає, що панель здатна генерувати енергію навіть у хмарні дні або при ранковому/вечірньому сонці. Вона максимізує виробництво енергії та забезпечує стабільну продуктивність у різних умовах.
4. Міцна та стійка конструкція: панель Trina Solar Bifacial 645W виготовлена з високоякісних матеріалів, які забезпечують її міцність та стійкість до механічних впливів, вітру та корозії. Вона може витримувати важкі погодні умови і має довгий термін служби.

5. Легка установка та сумісність: Trina Solar Bifacial 645W має просту та зручну систему монтажу, що дозволяє легко встановлювати її на різних поверхнях. Крім того, вона сумісна з різними системами монтажу та інверторами, що спрощує її інтеграцію в будь-яку сонячну енергетичну систему.

Make Sky Blue 60A контроллер

MPPT (Maximum Power Point Tracking) контролер "Make Sky Blue" - це пристрій, спеціально розроблений для оптимізації ефективності сонячних панелей шляхом пошуку та слідування за точкою максимальної потужності. Робота цього контролера може бути описана так [4]:

1. Відстеження точки максимальної потужності: MPPT контролер "Make Sky Blue" використовує алгоритми пошуку для визначення оптимальної робочої точки сонячних панелей, де вихідна потужність є найвищою. Це забезпечує максимальний вихідний струм і напругу для ефективного заряду акумуляторів або надання електроенергії споживачам.
2. Адаптивна регуляція параметрів: контролер автоматично адаптує свої параметри роботи в залежності від змін у рівні освітленості та температурі, щоб постійно забезпечувати оптимальну робочу точку панелей.
3. Ефективне заряджання акумуляторів: завдяки точному відстеженню максимальної потужності, контролер "Make Sky Blue" забезпечує ефективне та швидке заряджання акумуляторних батарей, що дозволяє зберігати отриману електроенергію для використання в періоди низької сонячної активності.
4. Захист від перенапруг та перевантажень: контролер вбудований з механізмами захисту, які дозволяють уникнути пошкодження сонячних панелей від перенапруг або перевантажень в електричних колах.

5. Компактний та енергоефективний дизайн: "Make Sky Blue" має компактну конструкцію та низьке споживання енергії, що робить його ідеальним вибором для різних сонячних систем з обмеженими ресурсами.

Загалом, MPPT контролер "Make Sky Blue" забезпечує оптимальну ефективність сонячних панелей та надійну роботу всієї сонячної системи, що робить його важливим компонентом в сучасних сонячних енергетичних установках.

Fchao KSC-3000W інвертор

Fchao KSC-3000W - це сонячний інвертор змінного струму (AC), призначений для конвертації постійного струму (DC), що генерується сонячними панелями, у високоякісний змінний струм, який може бути використаний для живлення побутових та комерційних електричних приладів [5].

Основна робота Fchao KSC-3000W може бути описана наступним чином:

1. Конвертація постійного струму в змінний: сонячні панелі генерують постійний струм (DC), який входить у внутрішній конвертер інвертора. Інвертор потім конвертує цей постійний струм у змінний струм (AC) зі стандартною частотою та напругою для використання в електричних системах будинків або комерційних приміщень.
2. Синхронізація з мережею: Fchao KSC-3000W може також синхронізувати вихідний змінний струм з мережею електропостачання, що дозволяє використовувати вироблену сонячну енергію для живлення електроустаткування в будинку або комерційному приміщенні.
3. Моніторинг та керування: деякі моделі Fchao KSC-3000W можуть мати вбудовані системи моніторингу та керування, які дозволяють

відстежувати ефективність сонячної системи та віддалено керувати роботою інвертора через мобільний додаток або веб-інтерфейс.

4. Захист від перенапруг: інвертори, такі як Fchao KSC-3000W, зазвичай мають вбудовані механізми захисту, які відключають пристрій від мережі в разі перенапруги або інших небезпечних умов роботи.
5. Ефективність та надійність: інвертори Fchao відомі своєю високою ефективністю та надійністю. Вони часто мають довгий термін служби та гарантію на якість виробу.

Загалом Fchao KSC-3000W є важливим компонентом сонячних енергетичних систем, який дозволяє ефективно використовувати електроенергію, що генерується сонячними панелями, для живлення різних електричних пристроїв і споживачів.

Arduino uno + WiFi плата

Arduino Uno з модулем WiFi може бути використаний для створення моніторингової та управляючої системи для автономної сонячної електростанції [6].

Загальний опис роботи такої системи:

1. Збір даних з сонячних панелей та акумуляторів: Arduino Uno може бути підключений до сенсорів, які вимірюють напругу та струм, що генеруються сонячними панелями, а також напругу та рівень заряду акумуляторів. Ці дані будуть збиратися і передаватися до мікроконтролера для подальшої обробки.
2. Передача даних через WiFi: Arduino Uno з модулем WiFi може використовуватися для передачі зібраних даних до хмарного сервісу або сервера для моніторингу в реальному часі. Це дозволяє власнику електростанції віддалено відстежувати її роботу та ефективність через мережу Інтернет.

3. Управління енергоефективністю: на основі отриманих даних Arduino Uno може виконувати аналіз ефективності роботи сонячної електростанції та регулювати робочі параметри, наприклад: кут нахилу сонячних панелей або режими заряду акумуляторів, для оптимізації використання сонячної енергії.
4. Сповіщення про стан системи: Arduino Uno може відправляти сповіщення власнику електростанції через WiFi у випадку виявлення проблем або аварійних ситуацій, наприклад: низького рівня заряду акумуляторів або несправності сонячних панелей.
5. Забезпечення безпеки та захисту: Arduino Uno може бути програмований для виконання заходів безпеки та захисту, наприклад: вимкнення електроенергії в разі перевищення певних параметрів чи виявлення небезпеки.

Така система дозволяє забезпечити моніторинг та управління автономною сонячною електростанцією з будь-якого місця, де є доступ до мережі Інтернет, що забезпечує зручність та ефективність її експлуатації.

Актуатор для сонячного трекера

Актуатор для сонячного трекера є ключовим компонентом системи, яка оптимізує положення сонячних панелей для максимального захоплення сонячної енергії. Такий актуатор дозволяє сонячним панелям слідкувати за рухом сонця протягом дня, що значно підвищує ефективність генерації електроенергії. Детальний огляд актуаторів для сонячних трекерів [7]:

Типи актуаторів для сонячних трекерів

1. **Лінійні актуатори:**
    - **Принцип дії:** Перетворюють обертальний рух електродвигуна в лінійний рух.
    - **Переваги:** Висока точність позиціонування, велика сила тяги, надійність.

- **Застосування:** Найчастіше використовуються у одноосьових трекерах для переміщення панелей по одній осі.

2. **Обертальні (ротаційні) актуатори:**
   - **Принцип дії:** Використовують електродвигун для створення обертального руху.
   - **Переваги:** Простота конструкції, можливість повороту на великі кути.
   - **Застосування:** Використовуються в двохосьових трекерах для забезпечення повороту панелей у двох площинах.

Основні характеристики

1. **Вантажопідйомність:** Визначає максимальну вагу, яку актуатор може переміщати. Це важливо для вибору актуатора, здатного витримати вагу сонячної панелі та конструкції трекера.
2. **Хід:** Відстань, яку може пройти лінійний актуатор від початкової до кінцевої точки.
3. **Швидкість:** Важлива характеристика для швидкого коригування положення панелей, особливо в умовах змінної хмарності.
4. **Точність позиціонування:** Визначає здатність актуатора точно розташовувати панель під необхідним кутом.
5. **Захист від погодних умов:** Оскільки сонячні трекери знаходяться на відкритому повітрі, актуатори повинні бути захищені від пилу, вологи та інших атмосферних впливів. Зазвичай це відображається у класі захисту IP.

Управління актуаторами

Актуатори для сонячних трекерів зазвичай керуються системою, яка включає:

1. **Контролер:** Центральний процесор, який обробляє дані з сенсорів та керує актуаторами.
2. **Сенсори:** Вимірюють положення сонця (фотодіоди, солярні датчики) та інші параметри (температура, вітер).
3. **Програмне забезпечення:** Алгоритми, що розраховують оптимальне положення панелей на основі даних від сенсорів.

Переваги використання актуаторів у сонячних трекерах

1. **Підвищення ефективності:** Дозволяє панелям слідкувати за сонцем, що збільшує виробництво електроенергії на 20-30% у порівнянні з нерухомими установками.
2. **Гнучкість установки:** Можливість встановлення в різних кліматичних умовах і місцях.
3. **Зниження витрат на електроенергію:** Більш ефективне використання панелей скорочує термін окупності системи.

Виклики та обмеження

1. **Вартість:** Початкові витрати на придбання та встановлення трекерів з актуаторами можуть бути вищими порівняно з фіксованими системами.
2. **Обслуговування:** Потребують регулярного технічного обслуговування для забезпечення довговічності та безперебійної роботи.
3. **Стійкість до погодних умов:** Необхідно враховувати можливість сильних вітрів, які можуть пошкодити систему, тому важливо вибирати актуатори з відповідними характеристиками міцності.

Таким чином, актуатори є критичним компонентом для сонячних трекерів, забезпечуючи їх ефективну роботу та підвищення продуктивності сонячних панелей.

1.4 Постановка задачі

Головною метою дослідження є розробка та впровадження скетчів Arduino, створення телеграм бота та локального веб-серверу, спрямованого на полегшення роботи з енергією виробленою сонячною панеллю. Сервер має надавати зручний та ефективний інтерфейс для отримання інформації про стан струму, доступ до керування розташуванням панелі, а також точне розташування відносно сонця, спрямовані на покращення кількості Ватт що виробляються панеллю. Телеграм бот матиме список команд для взаємодії з платою.

Для досягнення поставленої мети дослідження були сформульовані наступні завдання:

- аналіз потреб користувачів: провести аналіз потреб та вимог користувачів до функціоналу телеграм бота, спрямованого на полегшення слідкування за показниками та керування панеллю поза межами локальної мережі, локальний сервер для схожого функціоналу телеграм боту, але в локальній мережі.
- розробка архітектури контроллера arduino uno+WIFI: розробити архітектуру контроллера ESP8266 та ATmega328P, визначити їх основні функції та можливості.
- розробка інтерфейсу: створити зручний та інтуїтивно зрозумілий інтерфейс для взаємодії користувачів з панеллю як на локальному сервері, так і телеграм ботом.
- інтеграція систем спостереження: забезпечити інтеграцію телеграм бота та сервера з датчиками струму та напруги INA219 і фоторезисторами для спостереження за показниками.
- розробка функціоналу керування: розробити функціонал для керування, включаючи можливість зміни положення по горизонталі та вертикалі, як завдяки телеграм боту, так і локальному серверу.

- тестування та вдосконалення: провести тестування розроблених ресурсів для зручного керування, зібрати відгуки користувачів та внести необхідні виправлення та покращення.

Очікується, що результатом цього дослідження буде повнофункціональний телеграм бот, який забезпечить зручний доступ до стану генеруємого струму та акумуляторів, надаватиме можливість керувати положенням панелі, і, що найважливіше, робити це з будь-якої частини світу. Якщо користувач знаходиться в локальній мережі WIFI-маршрутизатора, то він може підключитись до мережі WIFI, перейти по IP-адресі локального серверу та спостерігати за станом приладів через смартфон. Таким чином можна в будь-який момент дізнатись стан струму та напруги, керувати конструкцією навіть не взаємодіючи з приладами.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Аналіз середовищ розробки

На Arduino UNO з модулем WiFi можна програмувати за допомогою різних інтегрованих середовищ розробки (IDE) та мов програмування. Наступні декілька популярних програм та мов, які можна використовувати для розробки програмного забезпечення для Arduino UNO з модулем WiFi:

1. **Arduino IDE:** Arduino IDE - це офіційне інтегроване середовище розробки для плат Arduino. Воно є безкоштовним та простим у використанні. Arduino IDE має вбудовану підтримку мови програмування C/C++, яка використовується для написання програм для Arduino. Arduino розробляє, виробляє та підтримує електронні пристрої та програмне забезпечення, що дозволяє людям у всьому світі легко отримувати доступ до передових технологій, які взаємодіють із фізичним світом. Їх продукти зрозумілі, прості та потужні, готові задовольнити потреби користувачів від студентів до творців і аж до професійних розробників. Для роботи з модулем WiFi потрібно встановити відповідну бібліотеку для роботи з мережевими функціями [8].
2. **PlatformIO:** PlatformIO - це відкрите інтегроване середовище розробки для різних мікроконтролерів, включаючи Arduino. Воно підтримує різні IDE, такі як Visual Studio Code, Atom і Eclipse.Унікальна філософія PlatformIO на ринку вбудованих пристроїв надає розробникам сучасне інтегроване середовище розробки (IDE для хмарних і настільних комп’ютерів), яке працює між платформами, підтримує багато різних комплектів розробки програмного забезпечення (SDK) або Frameworks і включає складне налагодження (Debugging), модульне тестування (Модульне тестування), автоматизований аналіз коду (Static Code Analysis) і віддалене керування (Remote Development). Він створений для максимальної гнучкості та можливості вибору для розробників, які

можуть використовувати або графічний редактор, або редактор командного рядка (PlatformIO Core (CLI)), або обидва [9].

3. **Node-RED:** Node-RED - це візуальне середовище програмування, яке дозволяє створювати програми шляхом перетягування та розміщення вузлів на віртуальній дошці. Воно підтримує роботу з Arduino через модуль ESP8266 або ESP32 як моста для комунікації з WiFi. Node-RED зазвичай використовується для швидкої розробки прототипів та Інтернету речей (IoT) [10].
4. **Python з використанням бібліотеки PySerial:** Python - це високорівнева мова програмування, яка також може бути використана для програмування Arduino через USB з використанням бібліотеки PySerial для взаємодії з мікроконтролером. Для використання з модулем WiFi можна використовувати відповідні бібліотеки Python для роботи з мережевими протоколами, такими як MQTT або HTTP [11].

Кожна з цих програм та мов має свої переваги та недоліки, і вибір залежить від особистого досвіду, потреб та вподобань.

2.2 Аналіз середовищ розробки

Для розробки скетчів була обрана програма Arduino IDE версії 2.3.2. Arduino використовує мову C++, має велику кількість бібліотек, функцій та справок, що забезпечують майже повну простоту в освоєнні, широку підтримку спільноти розробників, а також можливість створювати скетчі для різних потреб. Arduino IDE надає зручне середовище для програмування мікроконтролерів, що дозволяє швидко приступити до розробки простих і складних проектів [12].

Однією з ключових переваг Arduino є можливість використання технології OTA (Over-the-Air) для бездротового оновлення програмного забезпечення на мікроконтролерах, таких як ESP8266. Технологія OTA дозволяє віддалено завантажувати нові версії скетчів через бездротове з'єднання, що важливо для проектів, де доступ до мікроконтролера через USB-порт обмежений або недоцільний [13].

Підтримка штучного WIFI-порту для передачі скетчів забезпечує додаткову зручність та швидкість у випадках, коли використання USB-порту неможливе або неефективне. Передача через WIFI-порт набагато швидше, що забезпечує більш ефективне використання часу при розробці та оновленні програмного забезпечення.

Крім того, багатофункціональність Arduino IDE дозволяє розробникам легко взаємодіяти з мікроконтролерами та реалізувати різноманітні функції в їх проектах. Інтегроване середовище розробки має вбудовані інструменти для редагування коду, відладки та виконання скетчів, що робить процес розробки більш ефективним та зручним.

Додатково до використання Arduino IDE та технології OTA, для реалізації інтерфейсу користувача та веб-додатків було застосовано технології веб-розробки, такі як HTML, CSS і JavaScript. HTML використовується для створення структури веб-сторінок, CSS - для оформлення та стилізації цих елементів, а JavaScript - для реалізації інтерактивності та динамічної поведінки веб-додатків.

Інтеграція цих технологій дозволяє створювати зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що взаємодіють з мікроконтролером через веб-сервер. Використання HTML дозволяє створювати різноманітні елементи інтерфейсу, такі як: кнопки, поля введення і списки для управління різними функціями мікроконтролера. CSS додає можливості для оформлення та стилізації цих елементів, що покращує їх візуальний вигляд та зручність використання. JavaScript, у свою чергу, надає можливості для реалізації різноманітних функціональних можливостей, таких як: асинхронна обробка подій, валідація даних та динамічне оновлення веб-сторінок без перезавантаження [14].

Ці технології використовуються для створення зручного та ефективного інтерфейсу для користувачів, що спілкуються з мікроконтролером через веб-додаток. Вони доповнюють функціональність Arduino IDE та OTA, роблячи взаємодію з мікроконтролером ще більш зручною та привабливою для користувачів.

Для роботи з платою було обрано программу Arduino IDE. Вона надає доступ до вибору плат завдяки менеджеру плат, доступ до бібліотек (які також мають окремий менеджер) що допомагають в роботі з функціями скетчів, має доступ до монітору порту для перегляду даних що виводяться в Serial.

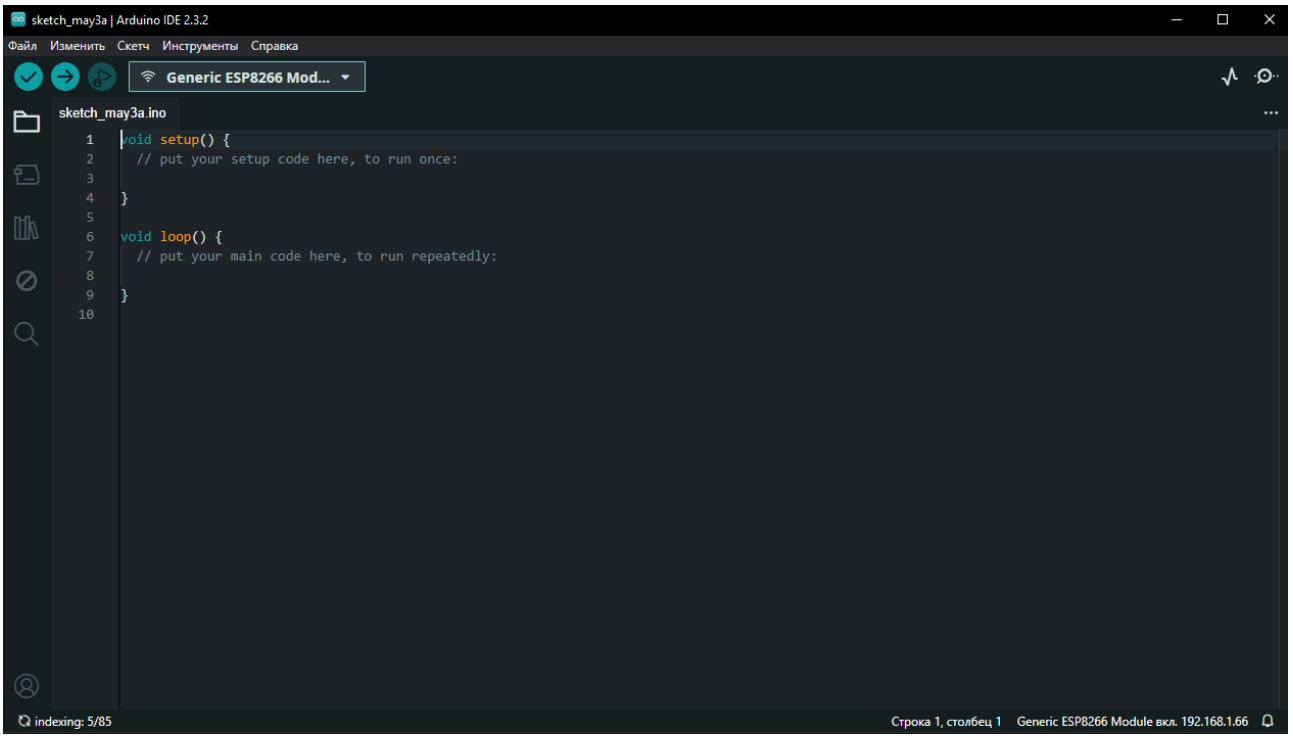

Рисунок 2.1 – Зовнішній інтерфейс програми Arduino IDE

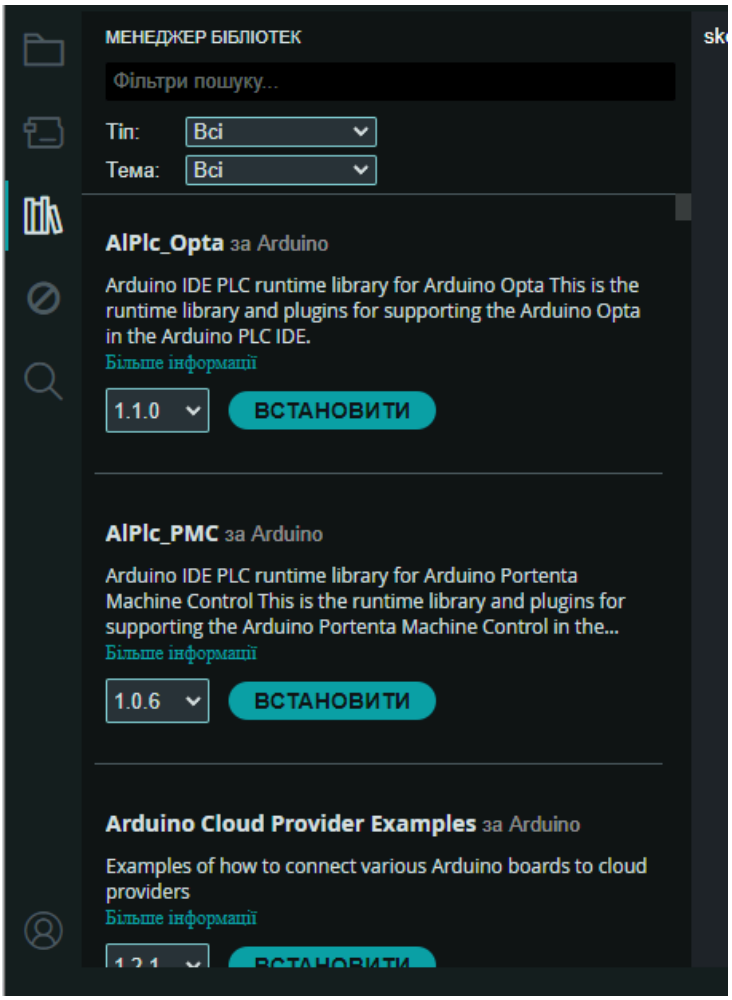

Рисунок 2.2 – Менеджер бібліотек Arduino IDE

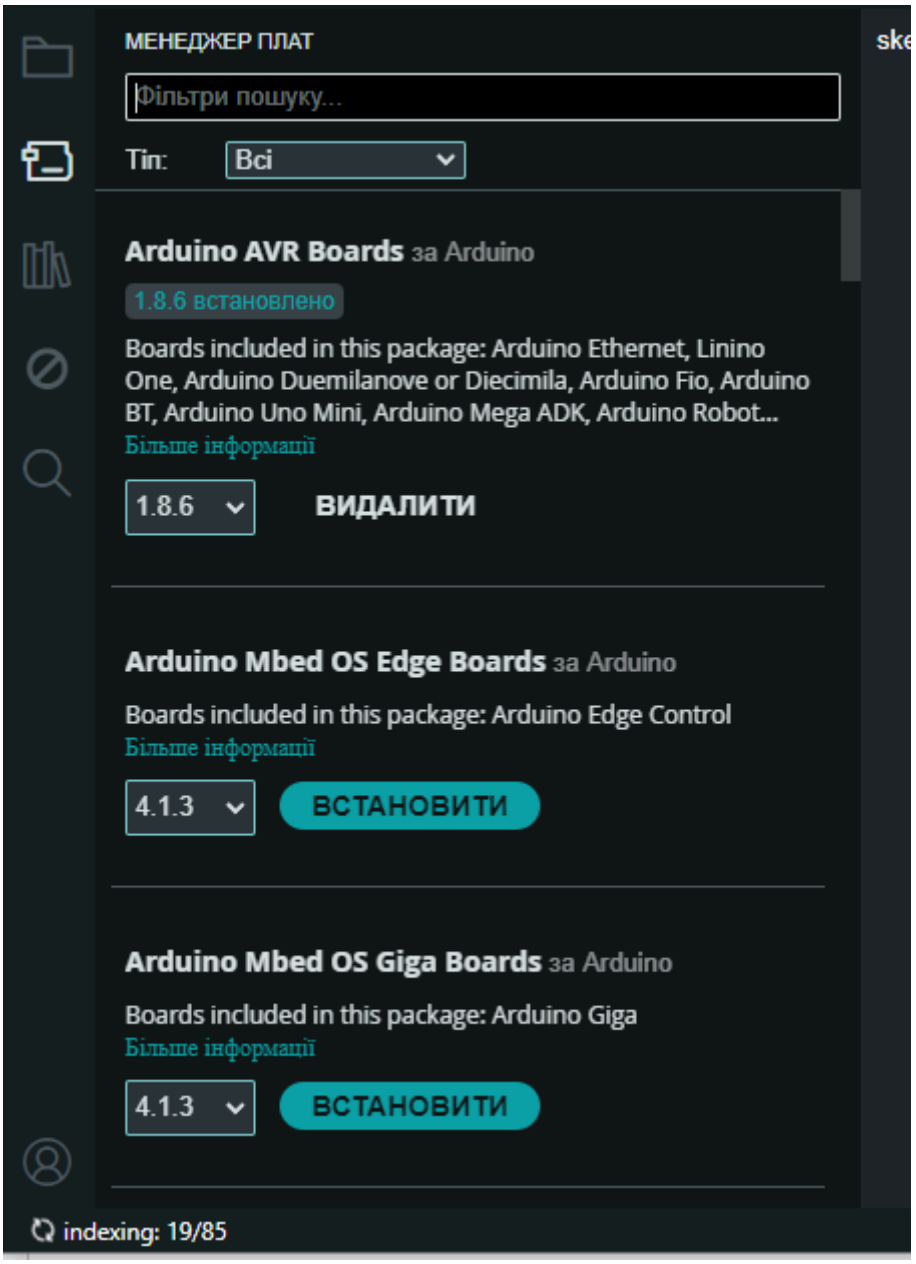

Рисунок 2.3 – Менеджер плат Arduino IDE

Arduino IDE постійно оновлюється додаючи нові можливості для програмування плат. Бібліотеки також удосконалюються для розширення функціоналу та можливостей майбутніх скетчів.

2.3 Інструменти створення ботів

Для створення Telegram ботів існує безліч інструментів, які можуть значно полегшити процес розробки. Одними з популярних є:

1. Telegram Bot API

Telegram Bot API - це офіційний API від Telegram для створення ботів. Він дозволяє розробникам взаємодіяти з Telegram серверами через HTTP-запити.

Основні можливості:

- Надсилання та отримання повідомлень.
- Обробка команд та повідомлень.
- Використання webhook для миттєвого отримання оновлень.
- Робота з різними типами медіа (фото, відео, аудіо, документи тощо).
- Підтримка inline-режиму.

2. BotFather

BotFather - це офіційний бот від Telegram, який допомагає створити та налаштувати нового бота. З його допомогою можна отримати токен для доступу до API та налаштувати основні параметри бота.

Основні можливості:

- Створення нового бота.
- Отримання токену для API.
- Налаштування опису, зображення профілю та команд бота.

3. Python-telegram-bot

Python-telegram-bot - це популярна бібліотека для Python, яка забезпечує зручний інтерфейс для роботи з Telegram Bot API.

Основні можливості:

- Простий у використанні синтаксис для взаємодії з API.
- Підтримка асинхронної обробки повідомлень.
- Легке налаштування webhook.
- Обробка різних типів повідомлень та команд.

4. Telebot (PyTelegramBotAPI)

Telebot - ще одна бібліотека для Python, яка дозволяє швидко створювати ботів для Telegram.

Основні можливості:

- Простий та зрозумілий інтерфейс.
- Підтримка різних типів повідомлень та команд.
- Легке налаштування webhook.

5. Node-telegram-bot-api

Node-telegram-bot-api - бібліотека для Node.js, яка забезпечує зручний інтерфейс для роботи з Telegram Bot API.

Основні можливості:

- Підтримка різних методів API Telegram.
- Підтримка як polling, так і webhook.
- Обробка команд та повідомлень.

6. Telegraf

Telegraf - це ще одна бібліотека для Node.js, яка надає розширені можливості для створення Telegram ботів.

Основні можливості:

- Легкий у використанні синтаксис.
- Підтримка middleware для обробки запитів.

- Підтримка різних методів API Telegram.
- Інтеграція з іншими сервісами та API.

7. Aiogram

Aiogram - асинхронна бібліотека для Python, яка використовує asyncio для високої продуктивності.

Основні можливості:

- Асинхронна обробка запитів.
- Підтримка webhook.
- Легка інтеграція з іншими асинхронними бібліотеками.

8. TDialogflow

Для створення ботів з підтримкою природної мови можна використовувати інтеграцію з Google Dialogflow.

Основні можливості:

- Розпізнавання та обробка природної мови.
- Підтримка інтеграції з Telegram через webhook.
- Налаштування складних діалогів та відповідей.

9. Microsoft Bot Framework

Microsoft Bot Framework - потужний інструмент для створення багатоплатформних ботів, включаючи Telegram.

Основні можливості:

- Підтримка створення ботів для різних платформ.
- Використання AI для обробки природної мови.
- Легке налаштування та розгортання.

Ці інструменти дозволяють створювати ботів з різним рівнем складності та функціональності. Вибір залежить від ваших конкретних потреб, знань програмування та переваг.

Для створення телеграм бота, що керує та надає інформацію з приладів вимірювання, був використаний телеграм бот BotFather та бібліотека UniversalTelegramBot. Він є офіційним інструментом для створення та налаштування ботів в Telegram. Дозволяє створювати, редагувати, керувати та видаляти ботів, а також налаштовувати їх параметри та функціонал.

Основні функції BotFather:

1. Створення бота: BotFather надає можливість створювати нових ботів шляхом надання їм унікальних імен. Крім того, він дозволяє встановлювати різні параметри, такі як ім'я, короткий опис, зображення профілю тощо.
2. Налаштування команд: BotFather дозволяє додавати та налаштовувати команди для бота. Це дозволяє користувачам спілкуватися з ботом та виконувати різні дії за допомогою спеціальних команд.
3. Управління доступом: BotFather дозволяє керувати доступом до бота, встановлюючи правила щодо того, хто може взаємодіяти з ним і які функції вони можуть використовувати.
4. Оновлення та видалення: BotFather дозволяє оновлювати і видаляти бота, якщо потрібно. Завдяки цьому можна виправити помилки, змінити параметри або повністю видалити його з месенджера.
5. Налаштування повідомлень: BotFather дозволяє налаштовувати повідомлення, які будуть відправлятися користувачам під час взаємодії з ботом. Отже можна налаштувати текст, зображення, відео та інші елементи повідомлення.
6. Документація та підтримка: BotFather надає доступ до документації та ресурсів, які допоможуть розуміти, як працює Telegram API та як краще використовувати його для створення ботів.

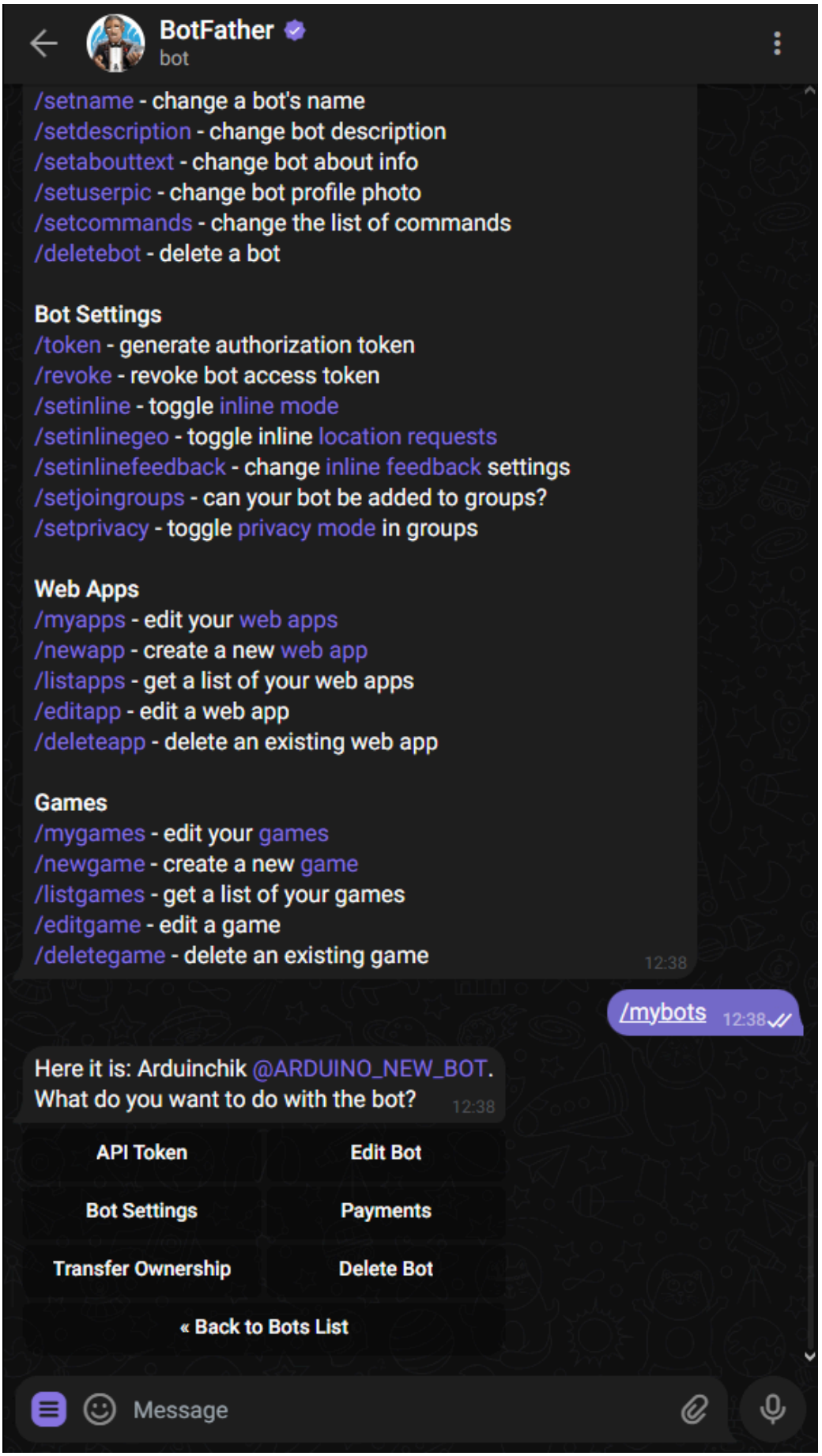

Рисунок 2.4 – Головний бот BotFather потрібен для створення ботів в телеграмі

Як працює BotFather:

1. Створення нового бота: почати взаємодію з BotFather можна надіславши йому команду /newbot та дотримуючись інструкцій для створення нового бота.
2. Налаштування параметрів бота: після створення бота можна налаштувати його параметри, такі як: ім'я, опис, зображення профілю та інші налаштування.
3. Додавання команд: для створюваного бота можна створити команди, які будуть виконувати певні дії при виклику.
4. Налаштування доступу: BotFather дозволяє налаштовувати правила доступу до нового бота, встановлюючи параметри конфіденційності та безпеки.
5. Завершення налаштувань: після того як всі параметри бота налаштовані, BotFather надає API-ключ та іншу необхідну інформацію для роботи з новим ботом.

BotFather є важливим інструментом для розробки ботів в Telegram, який дозволяє створювати та налаштовувати ботів швидко і ефективно.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ Й ТЕСТУВАННЯ

3.1 ПЗ для розробки программ

Для реалізації функціоналу плати були використані наступні бібліотеки:

1. ESP8266WiFi.h

Ця бібліотека забезпечує основну підтримку Wi-Fi для мікроконтролерів ESP8266. Вона включає функції для підключення до Wi-Fi мережі, управління з'єднаннями та отримання інформації про мережу.

Основні функції:

- WiFi.begin(ssid, password) — підключення до Wi-Fi мережі.
- WiFi.status() — перевірка стану з'єднання.
- WiFi.localIP() — отримання IP-адреси пристрою.
- WiFi.disconnect() — відключення від мережі.

2. ESP8266WebServer.h

Ця бібліотека дозволяє створювати прості веб-сервери на базі ESP8266. Вона дає можливість обробляти HTTP-запити, надсилати відповіді та керувати ресурсами.

Основні функції:

- server.on(path, HTTPMethod, handler) — визначення обробника для конкретного шляху та методу HTTP.
- server.begin() — запуск веб-сервера.
- server.handleClient() — обробка вхідних запитів.

3. WiFiClientSecure.h

Ця бібліотека додає підтримку SSL/TLS для захищених з'єднань через Wi-Fi. Вона базується на WiFiClient і дозволяє створювати безпечні клієнтські з'єднання для передачі конфіденційних даних.

Основні функції:

- WiFiClientSecure.connect(host, port) — встановлення захищеного з'єднання з сервером.
- WiFiClientSecure.print(data) — відправка даних через захищене з'єднання.
- WiFiClientSecure.read() — зчитування даних з захищеного з'єднання.

4. UniversalTelegramBot.h

Ця бібліотека дозволяє взаємодіяти з API Telegram, що дає змогу створювати ботів на базі ESP8266. Вона спрощує відправку та отримання повідомлень через ботів.

Основні функції:

- UniversalTelegramBot.sendMessage(chat_id, text, parse_mode) — відправка повідомлення користувачеві або групі.
- UniversalTelegramBot.getUpdates(bot_last_update) — отримання нових повідомлень.
- UniversalTelegramBot.answerCallbackQuery(callback_query_id, text, show_alert) — відповідь на запити з інлайн-кнопок.

5. ESP8266mDNS.h

Ця бібліотека забезпечує підтримку Multicast DNS (mDNS), що дозволяє знаходити пристрої в локальній мережі за іменами без необхідності знати їх IP-адреси.

Основні функції:

- MDNS.begin(hostname) — запуск mDNS служби з заданим іменем хоста.

- MDNS.addService(service, protocol, port) — додавання сервісу для оголошення через mDNS.
- MDNS.queryService(service, protocol) — пошук сервісів в мережі.

6. WiFiUdp.h

Ця бібліотека додає підтримку UDP (User Datagram Protocol) для обміну даними в локальній мережі. Вона підходить для задач, де потрібна висока швидкість і не критична надійність передачі.

Основні функції:

- WiFiUDP.begin(port) — запуск UDP сервера на заданому порту.
- WiFiUDP.beginPacket(host, port) — підготовка до відправки UDP пакету.
- WiFiUDP.write(data) — запис даних в UDP пакет.
- WiFiUDP.endPacket() — відправка UDP пакету.
- WiFiUDP.parsePacket() — перевірка наявності вхідного UDP пакету.
- WiFiUDP.read() — зчитування даних з UDP пакету.

7. ArduinoOTA.h

Ця бібліотека дозволяє оновлювати прошивку ESP8266 через Wi-Fi (Over-the-Air, OTA). Це дуже зручно для віддаленого оновлення пристроїв без фізичного доступу до них.

Основні функції:

- ArduinoOTA.begin() — ініціалізація OTA.
- ArduinoOTA.handle() — обробка OTA запитів (повинна викликатися регулярно в головному циклі програми).
- ArduinoOTA.onStart(callback) — налаштування callback функції, яка виконується на початку OTA оновлення.
- ArduinoOTA.onEnd(callback) — налаштування callback функції, яка виконується після завершення OTA оновлення.

- ArduinoOTA.onProgress(callback) — налаштування callback функції, яка виконується під час OTA оновлення (для відображення прогресу).
- ArduinoOTA.onError(callback) — налаштування callback функції, яка виконується у разі помилки під час OTA оновлення.

8. Wire.h

Ця бібліотека забезпечує підтримку I2C (Inter-Integrated Circuit) протоколу, який використовується для комунікації між мікроконтролерами і периферійними пристроями, такими як датчики, дисплеї і т.д.

Основні функції:

- Wire.begin() — ініціалізація I2C-шини.
- Wire.beginTransmission(address) — початок передачі даних на пристрій з заданою адресою.
- Wire.write(data) — відправка даних на I2C-шину.
- Wire.endTransmission() — завершення передачі даних.
- Wire.requestFrom(address, quantity) — запит даних від пристрою з заданою адресою.
- Wire.read() — зчитування байта даних з I2C-шини.

9. Adafruit_INA219.h

Ця бібліотека призначена для роботи з датчиком струму INA219 від Adafruit, який дозволяє вимірювати напругу і струм в електричних ланцюгах. Вона використовує I2C для комунікації з мікроконтролером.

Основні функції:

- Adafruit_INA219.begin() — ініціалізація датчика INA219.
- Adafruit_INA219.getBusVoltage_V() — отримання напруги на шині (у вольтах).

- Adafruit_INA219.getShuntVoltage_mV() — отримання напруги на шунті (у мілівольтах).
- Adafruit_INA219.getCurrent_mA() — отримання значення струму (у міліамперах).
- Adafruit_INA219.getPower_mW() — отримання значення потужності (у міліватах).
- Adafruit_INA219.setCalibration_xxx() — налаштування калібрування датчика для різних режимів вимірювань.

Бібліотеки є одним із найважливіших елементів коду, без яких виклик потрібних функцій буде неможливим, або некоректно працюючим. Вони постійно оновлюються розробниками, заміняються, поповнюються аддонами для більш вузького напряму програмування.

3.2 Розробка програмного забезпечення для плат

Було розроблено програму для ESP8266, яка створює веб-сервер, що відображає дані із певних сенсорів та дозволяє керувати певними пристроями через веб-інтерфейс. Деякі ключові елементи коду:

- Підключення бібліотек: Для взаємодії з ESP8266 та Telegram Bot API використовуються різні бібліотеки, такі як ESP8266WiFi.h, ESP8266WebServer.h, WiFiClientSecure.h, UniversalTelegramBot.h, ESP8266mDNS.h, та WiFiUdp.h.
- Константи та змінні: Тут оголошені ключі Telegram Bot API, SSID та пароль для підключення до WiFi, а також IP-адреси для точки доступу (AP) та інші змінні для збереження даних сенсорів.

```
#define BOT_TOKEN "TOKEN"
#define CHAT_ID "ID"

IPAddress apIP(192, 168, 4, 1);
String _ssid = "SSID";
String _password = "1234567890";
String _ssidAP = "Arduino";
String _passwordAP = "";
```

Рисунок 3.1 – Константи та змінні

- Ініціалізація WiFi: Функція WIFIinit() встановлює підключення до WiFi мережі або, у випадку відсутності з'єднання, переходить до режиму точки доступу (AP).

```
void WIFIinit() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(_ssid.c_str(), _password.c_str());

  int tries = 0;
  while (WiFi.waitForConnectResult() != WL_CONNECTED && tries < 10) {
    ++tries;
    delay(1000);
    Serial.print(".");
  }

  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("");
    Serial.println("WiFi up AP");
    StartAPMode();
  } else {
    Serial.println("");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
  }
  ArduinoOTA.setHostname("ESP8266-panel");
  ArduinoOTA.setPassword("123");
  ArduinoOTA.begin();
}
```

Рисунок 3.2 – Функція WIFIinit

- Веб-сервер: У функції Server() налаштовується веб-сервер та визначаються обробники для основної сторінки, запитів на отримання даних та кнопок для керування.

```
void Server() {
  Serial.begin(115200);

  if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(_ssid.c_str(), _password.c_str());
    while (WiFi.waitForConnectResult() != WL_CONNECTED) {
      delay(1000);
      Serial.println("Connecting to WiFi...");
    }
    Serial.println(WiFi.localIP());
  }

  server.on("/", handleRoot);
  server.on("/North", handleN);
  server.on("/East", handleE);
  server.on("/South", handleS);
  server.on("/West", handleW);
  server.on("/data", handleDataRequest);
  server.begin();
}
```

Рисунок 3.3 – Функція Server

- Обробники запитів: Функції handleRoot(), handleN(), handleE(), handleS(), handleW(), handleDataRequest() відповідають за обробку відповідних запитів від клієнта.

```
void handleN() {
  Serial.write('7');
  bot.sendMessage(CHAT_ID, "Вгору");
}

void handleE() {
  Serial.write('5');
  bot.sendMessage(CHAT_ID, "Праворуч");
}
void handleS() {
  Serial.write('8');
  bot.sendMessage(CHAT_ID, "Вниз");
}

void handleW() {
  Serial.write('4');
  bot.sendMessage(CHAT_ID, "Ліворуч");
}
```

Рисунок 3.4 – Функції обробки

- Прослуховування серійного порту: У головному циклі loop() здійснюється постійне прослуховування серійного порту для отримання даних від сенсорів або віддалених команд.

```
void loop() {
  server.handleClient();
  while (Serial.available() > 0) {
    char dataType = Serial.read();
    if (dataType == 'V') {
      char sensorIndex = Serial.read();
      busVoltage[sensorIndex - '0'] = Serial.parseFloat();
    } else if (dataType == 'A') {
      char sensorIndex = Serial.read();
      current[sensorIndex - '0'] = Serial.parseFloat();
    } else if (dataType == 'N') {
      lightSensor = Serial.parseInt();
    } else if (dataType == 'S') {
      lightSensor1 = Serial.parseInt();
    } else if (dataType == 'E') {
      lightSensor2 = Serial.parseInt();
    } else if (dataType == 'W') {
      lightSensor3 = Serial.parseInt();
    }
  }
```

Рисунок 3.5 – Прослуховування серійного порту

- Взаємодія з Telegram: За допомогою Telegram Bot API бот може надсилати та отримувати повідомлення. Функція handleNewMessages() обробляє вхідні повідомлення.

```
for (int i = 0; i < numNewMessages; i++) {
  String chat_id = String(bot.messages[i].chat_id);
  String text = bot.messages[i].text;
  if (text.equals("Status")) {
    String message = "Bus Voltage 1: " + String(busVoltage[1]) + " V\n";
    message += "Current 1: " + String(current[1]) + " A\n";
    message += "Bus Voltage 2: " + String(busVoltage[2]) + " V\n";
    message += "Current 2: " + String(current[2]) + " A\n";
    message += "North: " + String(lightSensor) + "\n";
    message += "West: " + String(lightSensor1) + "\n";
    message += "East: " + String(lightSensor2) + "\n";
    message += "South: " + String(lightSensor3) + "\n";
    bot.sendMessage(chat_id, message);
```

Рисунок 3.6 – Зчитування даних телеграм ботом

- OTA оновлення: Використовується Arduino OTA для оновлення прошивки пристрою через мережу WiFi без необхідності підключення по USB.

Окрім неї була створена програма для ATmega328P, яка використовує два датчики тока INA219 та чотири фоторезистори для вимірювання напруги та струму, а також освітленості на певних напрямках (північ, південь, захід, схід). Основні елементи цього коду:

- Підключення бібліотек: Використовується бібліотека Wire.h для забезпечення комунікації по шині I2C з датчиками INA219.
- Оголошення змінних: Оголошуються об'єкти для кожного з датчиків тока INA219, а також змінні для збереження значень освітленості та інші допоміжні змінні.

```
#define INA219_ADDRESS_1 0x40
#define INA219_ADDRESS_2 0x44

Adafruit_INA219 ina1(INA219_ADDRESS_1);
Adafruit_INA219 ina2(INA219_ADDRESS_2);

int lightSensor = 0;
int lightSensor1 = 0;
int lightSensor2 = 0;
int lightSensor3 = 0;
int lightsensors;
int printCounter = 0;

unsigned long startTime = 0;
const unsigned long activeTime = 3000;
```

Рисунок 3.7 – Оголошення змінних

- Ініціалізація: У функції setup() налаштовуються піни введення/виведення та розпочинається серійне з'єднання.

```
void setup() {
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(115200);
  Wire.begin();
  ina1.begin();
  ina2.begin();
}
```

Рисунок 3.8 – Ініціалізація

- Цикл виконання: Основна логіка виконання програми розміщена у функції loop(). Програма чекає наявність даних у порті введення/виведення (серійний порт), і якщо вони є, виконує відповідні дії в залежності від отриманих команд (наприклад, включення/вимкнення світла).

```
void loop() {
  if (Serial.available() > 0) {
    char inChar = Serial.read();
    if (inChar == '7') {
  digitalWrite(7, LOW);
  startTime = millis();
} else if (inChar == '5') {
  digitalWrite(13, LOW);
  startTime = millis();
} else if (inChar == '8') {
  digitalWrite(8, LOW);
  startTime = millis();
} else if (inChar == '4') {
  digitalWrite(12, LOW);
  startTime = millis();
}
  }
```

Рисунок 3.9 – Функція з циклами виконання

- Вимірювання даних: За допомогою датчиків INA219 та аналогових вимірювальних пінів здійснюється вимірювання напруги, струму та освітленості. Отримані дані відправляються через серійний порт.

```
Serial.print('V');
Serial.print('1');
Serial.print(ina1.getBusVoltage_V());
Serial.print('A');
Serial.print('1');
Serial.print(ina1.getCurrent_mA());

Serial.print('V');
Serial.print('2');
Serial.print(ina2.getBusVoltage_V());
Serial.print('A');
Serial.print('2');
Serial.println(ina2.getCurrent_mA());
```

Рисунок 3.10 – Вивід даних через Serial порт

- Обробка команд: Програма читає отримані показники з фоторезисторів, перетворює їх в читабельний формат для С і відповідно до них відправляє дані через серійний порт.

```
if (Serial.available()> 0) {
  lightsensors = Serial.parseInt();
}
```

Рисунок 3.11 – Зчитування показників фоторезисторів

- Затримка та очищення буфера: Використовується затримка для забезпечення певного інтервалу між вимірюваннями, а також очищення буфера введення/виведення для запобігання переповненню.

```
delay(1000);
Serial.setTimeout(1000);
```

Рисунок 3.12 – Затримка та очищення буфера

- Періодичне відправлення даних: Для запобігання перевантаженню виводу дані відправляються через серійний порт не частіше, ніж кожні 10 ітерацій циклу виконання.

```
printCounter++;

if (printCounter >= 10) {
  while (Serial.available()) {
    Serial.read();
  }
  printCounter = 0;
}
```

Рисунок 3.13 – Зменшення навантаження на плату

Два вищеописаних коди взаємодіють один з одним. ATmega328P зчитує дані з датчиків INA219 разом з показниками фоторезисторів, завдяки технології I2C він транспортує їх до ESP8266 яка сканує Serial порт на рахунок нових даних. Сама ж ESP створює локальний веб сервер на статичній IP-адресі та відправляє туди дані з показників. При спробі користувачем дізнатись показники через телеграм бота вона обирає останні та передає їх через повідомлення серверами телеграм, що дозволяє дізнатись про стан системи в будь-який момент.

3.3 Тестування системи

Використовуючи інструмент BotFather та бібліотеку UniversalTelegramBot був створений телеграм бот “Arduinchik”. Бот має різноманітний функціонал та команди, які надають користувачам зручний інструмент для отримання необхідної інформації та взаємодії з ботом.

До найважливіших команд бота включаються:

- Status: команда що відображає показники вольтметрів, амперметрів та фоторезисторів
- W: Зміна напряму панелі на Захід
- S: Зміна напряму панелі на Південь
- E: Зміна напряму панелі на Схід
- N: Зміна напряму панелі на Північ

Усі ці команди та функції бота спрямовані на зручне керування сонячною панеллю за межами локальної мережі.

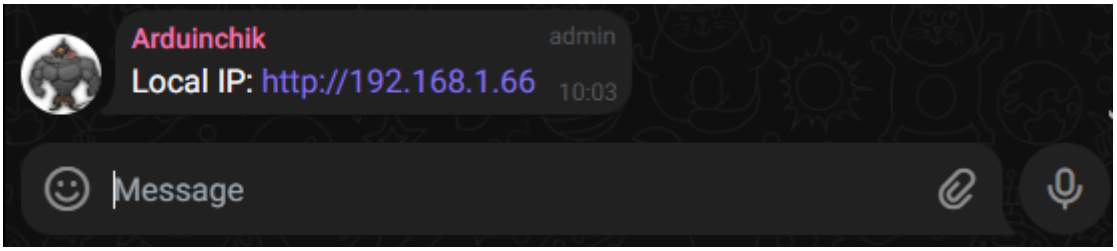

Рисунок 3.14 – Початок взаємодії з botom Arduinchik в телеграмі.

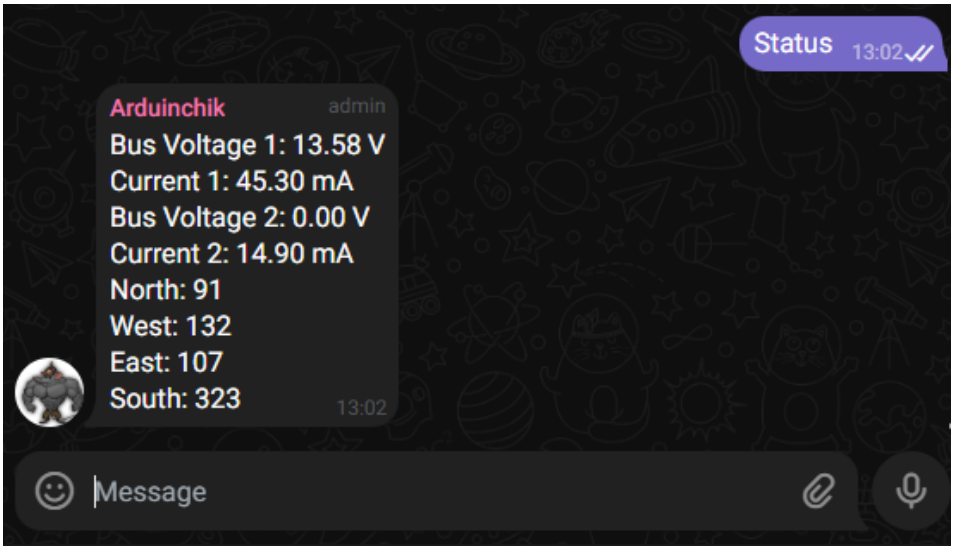

Рисунок 3.15 – Демонстрація команди Status

Рисунок 3.16 – Демонстрація команди N

Рисунок 3.17 – Демонстрація команди E

Рисунок 3.18 – Демонстрація команди S

Рисунок 3.19 – Демонстрація команди W

Також присутній локальний сервер на якому зручно відображаються всі необхідні показники на приладах.

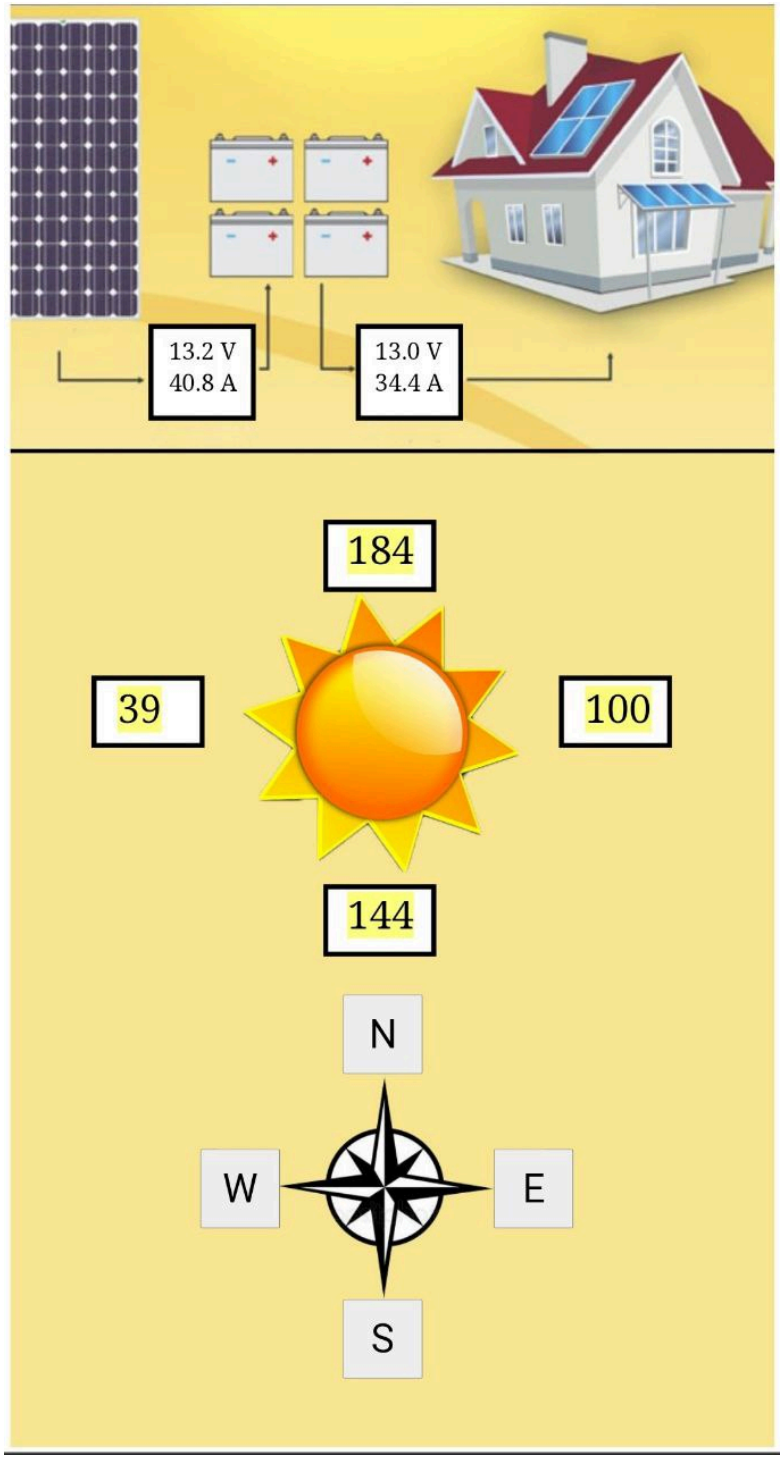

Рисунок 3.20 – Зовнішній вигляд локального серверу

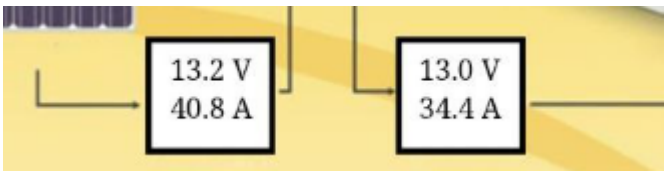

Рисунок 3.21 – Показники вольтметрів та амперметрів

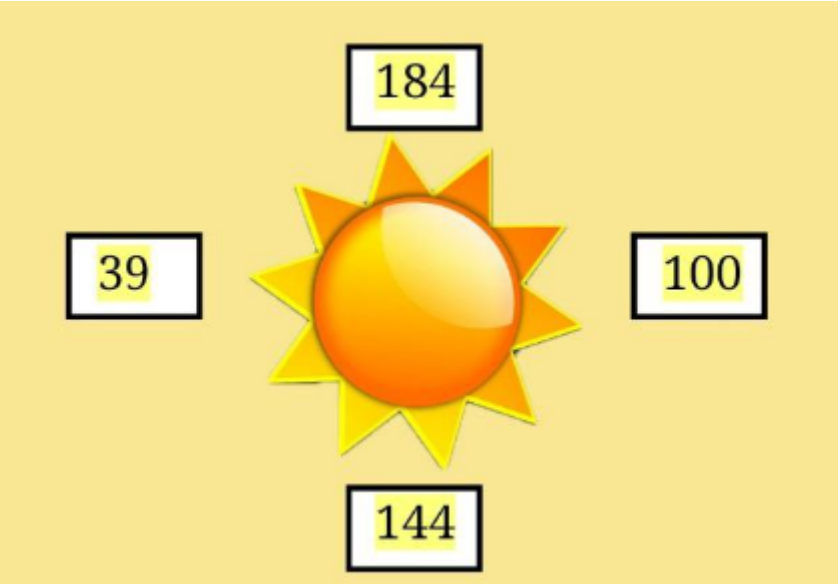

Рисунок 3.22 – Показники фоторезисторів

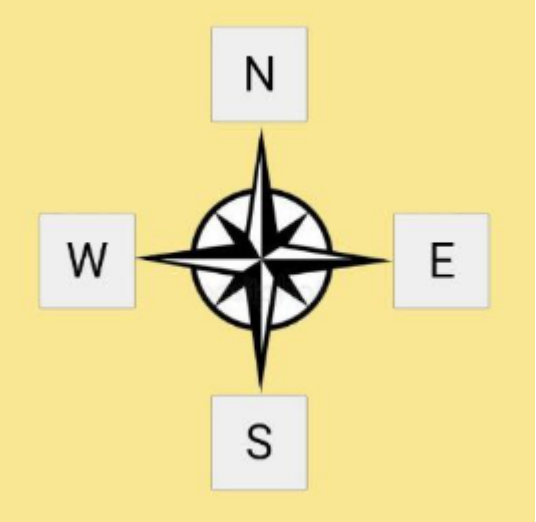

Рисунок 3.23 – Кнопки керування положенням панелі

Всі показники вольтметрів і амперметрів зчитуються з датчиків INA219 та прописуються в моніторі порту плати ESP8266, після чого виводяться в поля представлені на рисунку 3.21. Показники фоторезисторів аналогічним методом відображаються в моніторі порту і виводятся на рисунку 3.22. Програмний код представлений у додатку.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було проведено аналіз сучасних підходів до керування системою стеження за сонцем. Визначено важливість контролю напруги що виробляється та її використання, розглянуто методи керування для ефективної роботи протягом доби.

У ході виконання кваліфікаційної роботи бакалавра було виконано наступні завдання:

1. Аналіз сучасних методів моніторингу та управління системами стеження.
2. Розроблено та налаштовано скрипти для зчитування інформації з датчиків та керування положення панелі відносно сонця, що забезпечує максимальну користь з сонячних променів.
3. Налаштовано окрему статичну IP-адресу для ESP8266.
4. Проведено інтеграцію OTA для зручного завантаження налаштувань без прямого підключення до плати.
5. Створено локальний Web-сервер для моніторингу та керування панеллю в локальній мережі.
6. Створено телеграм бота з аналогічними функціями локального серверу, але доступ до даних можливий навіть поза межами локальної мережі.

Отже, можна зробити висновок, що розроблена система стеження за сонцем з можливістю віддаленого редагування налаштувань та керування є ефективним та перспективним рішенням в області зеленої енергетики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Soncedim [Електроний ресурс] – Режим доступу до ресурсу: https://soncedim.com.ua/blog/plyusy-ta-minusy-sonyachnyh-batarej (дата звернення: 08.04.2024).

2. Atmosfera [Електроний ресурс] – Режим доступу до ресурсу: https://www.atmosfera.ua/media/tipi-sonyachnih-elektrostancij?utm_source=google&utm_content=cid|20643381977|gid||kwid|&gad_source=1&gclid=CjwKCAjw9IayBhBJEiwAVuc3fkjNznyuRzMc-lCkwpZiUFlO4HvNCErXraxXJHBI3PKFw8ncDdpqQxoCAAIQAvD_BwE (дата звернення: 08.04.2024).

3. Brilliantsolar [Електроний ресурс] – Режим доступу до ресурсу: https://brilliantsolar.com.ua/ua/p1502156736-trina-solar-bifacial.html (дата звернення: 08.04.2024).

4. Solar-tech [Електроний ресурс] – https://solar-tech.com.ua/ua/battery-charge-controllers/solnechnyi-kontroller-zaryada-logicpower-makeskyblue-mppt-48v-60a-v120.html (дата звернення: 08.04.2024).

5. Fchao [Електроний ресурс] – Режим доступу до ресурсу: https://fchao.com.ua/uk/KSC-3000W (дата звернення: 08.04.2024).

6. Arduino [Електроний ресурс] – Режим доступу до ресурсу: https://arduino.ua/prod2014-plata-razrabotchika-arduino-unowifi-r3 (дата звернення: 09.04.2024).

7. Сонячні електростанції на трекерах [Електроний ресурс] – Режим доступу до ресурсу: https://setech.in.ua/sonachni-elektrostantsiyi-na-trekerah/ (дата звернення: 09.04.2024).

8. Software|Arduino [Електроний ресурс] – Режим доступу до ресурсу: https://www.arduino.cc/en/software (дата звернення: 09.04.2024).

9. Gateway to Embedded Software Development Excellence [Електроний ресурс] – Режим доступу до ресурсу: https://docs.platformio.org/en/latest/ (дата звернення: 09.04.2024).

10. Node-RED|Interacting with Arduino [Електроний ресурс] – Режим

доступу до ресурсу: https://nodered.org/docs/faq/interacting-with-arduino (дата звернення: 09.04.2024).

11. PySerial [Електроний ресурс] – Режим доступу до ресурсу: https://pyserial.readthedocs.io/en/latest/ (дата звернення: 10.04.2024).

12. Arduino [Електроний ресурс] – Режим доступу до ресурсу: https://www.arduino.cc (дата звернення: 10.04.2024).

13. The EASY Guide To Over-The-Air (OTA) Updates With ArduinoOTA [Електроний ресурс] – Режим доступу до ресурсу: https://youtu.be/7Xdsc1qqoro?si=yJna8RsOLDabIo7d (дата звернення: 10.04.2024).

14. HTML і CSS довідник українською [Електроний ресурс] – Режим доступу до ресурсу: https://html-css.co.ua (дата звернення: 13.04.2024).

15. How to Assign a Static / Fixed IP Address | ESP8266 | ESP32 | Arduino [Електроний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=B9jJI7p2Gw4 (дата звернення: 13.04.2024).

16. How to Setup Arduino UNO WiFi R3 ATmega328P ESP8266 [Електроний ресурс] – Режим доступу до ресурсу: https://youtu.be/K7h1w4v63N8?si=or9s2D1tQZzhGkHv (дата звернення: 15.04.2024).

17. HOW TO CONNECT ESP8266 TO WIFI NETWORK | Ut Go [Електроний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=fMjHY1m4nkQ (дата звернення: 15.04.2024).

18. Build an ESP8266 Web Server with Arduino IDE - Code and Schematics [Електроний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=dWM4p_KaTHY (дата звернення: 17.04.2024).

19. INA219 Tutorial(Arduino) [Електроний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=A8KjEubjhdo (дата звернення: 17.04.2024).

20. ATmega328P [Електроний ресурс] – Режим доступу до ресурсу:

https://content.arduino.cc/assets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (дата звернення: 18.04.2024).

21. Combine Use ATmega328P+ESP8266 with Blynk ioT | Communication between ATmega328P and ESP8266 | Hindi [Електроний ресурс] – Режим доступу до ресурсу: https://youtu.be/CvNrY1tF4Mg?si=7aFz31d_Ld1iwKoi (дата звернення: 18.04.2024).

22. UNO WiFi [Електроний ресурс] – Режим доступу до ресурсу: https://docs.arduino.cc/retired/getting-started-guides/ArduinoUnoWiFi/ (дата звернення: 20.04.2024).

23. Arduino UNO WIFI board [Електроний ресурс] – Режим доступу до ресурсу: https://docs.arduino.cc/retired/boards/arduino-uno-wifi/ (дата звернення: 20.04.2024).

24. LittleFS [Електроний ресурс] – Режим доступу до ресурсу: https://github.com/esp8266/Arduino/tree/master/libraries/LittleFS/src (дата звернення: 22.04.2024).

25. ESP8266 Telegram Home Automation Feedback System [Електроний ресурс] – Режим доступу до ресурсу: https://youtu.be/GcMTlD8ymUM?si=nWqk-gIzsnEe2ONC (дата звернення: 23.04.2024).

26. How to upload real-time data on Webserver? [Електроний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=Jc-o6XFlv_E (дата звернення: 25.04.2024).

27. NodeMCU ESP8266 | Tutorial via Arduino IDE [Електроний ресурс] – Режим доступу до ресурсу: https://youtu.be/NmkeIpD8WG8?si=t6FMGBzm_Jvn4yxC (дата звернення: 25.04.2024).

28. Solar Tracking Systems [Електроний ресурс] – Режим доступу до ресурсу:https://www.slideshare.net/ArupendraGhosh/solar-tracking-systems?next_slideshow=true (дата звернення: 28.04.2024).

Додаток А

ESP8266:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>

#define BOT_TOKEN "TOKEN"
#define CHAT_ID "ID"

IPAddress apIP(192, 168, 4, 1);
String _ssid = "SSID";
String _password = "123";
String _ssidAP = "Arduino";
String _passwordAP = "";

ESP8266WebServer server(80);
WiFiClientSecure client;
const unsigned long BOT_MTBS = 1000;
X509List cert(TELEGRAM_CERTIFICATE_ROOT);
UniversalTelegramBot bot(BOT_TOKEN, client);
unsigned long bot_lasttime;

float busVoltage[4];
float current[4];
int lightSensor, lightSensor1, lightSensor2, lightSensor3;

void WIFIinit() {
  WiFi.mode(WIFI_STA);
```

```
 WiFi.begin(_ssid.c_str(), _password.c_str());

 int tries = 0;
 while (WiFi.waitForConnectResult() != WL_CONNECTED && tries < 10) {
  ++tries;
  delay(1000);
  Serial.print(".");
 }

 if (WiFi.status() != WL_CONNECTED) {
  Serial.println("");
  Serial.println("WiFi up AP");
  StartAPMode();
 } else {
  Serial.println("");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
 }
 ArduinoOTA.setHostname("ESP8266-panel");
 ArduinoOTA.setPassword("123");
 ArduinoOTA.begin();
}

bool StartAPMode() {
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
 WiFi.softAP(_ssidAP.c_str(), _passwordAP.c_str());
 return true;
}

void handleRoot() {
 String html = "<html><head><script>";
```

```
 html += "setInterval(function() {";
 html += "var xhr = new XMLHttpRequest();";
 html += "xhr.open('GET', '/data', true);";
 html += "xhr.onreadystatechange = function() {";
 html += "if (xhr.readyState == 4 && xhr.status == 200) {";
 html += "var data = xhr.responseText.split(';');";
            html      +=      "document.getElementById('busVoltage').innerText      =
parseFloat(data[0]).toFixed(1) +' V';";
  html += "document.getElementById('current').innerText = (parseFloat(data[1]) *
1.25).toFixed(1) +' A';";
            html      +=      "document.getElementById('busVoltage1').innerText      =
parseFloat(data[2]).toFixed(1) +' V';";
  html += "document.getElementById('current1').innerText = (parseFloat(data[3]) *
1.25).toFixed(1) +' A';";
 html += "document.getElementById('lightSensor').innerText = data[4];";
 html += "document.getElementById('lightSensor1').innerText = data[5];";
 html += "document.getElementById('lightSensor2').innerText = data[6];";
 html += "document.getElementById('lightSensor3').innerText = data[7];";
 html += "}";
 html += "};";
 html += "xhr.send();";
 html += "}, 300);";
 html += "</script></head><body>";
 html += "<style>";
 html += "#backgroundBlock {";
                    html          +=          "          background-image:
url('https://raw.githubusercontent.com/Username123542765/panel/main/fone.png');";
 html += "  background-size: cover;";
 html += "  background-position: center;";
 html += "  width: 100%;";
 html += "  height: 100%;";
 html += "  position: relative;";
 html += "}";
```

```
html += "#busVoltage {";
html += "  position: absolute;";
html += "  margin-top: 420px;";
html += "  margin-left: 200px;";
html += "  font-size: 30px;";
html += "}";
html += "#current {";
html += "  position: absolute;";
html += "  margin-top: 460px;";
html += "  margin-left: 180px;";
html += "  font-size: 30px;";
html += "}";
html += "#busVoltage1 {";
html += "  position: absolute;";
html += "  margin-top: 420px;";
html += "  margin-left: 460px;";
html += "  font-size: 30px;";
html += "}";
html += "#current1 {";
html += "  position: absolute;";
html += "  margin-top: 460px;";
html += "  margin-left: 440px;";
html += "  font-size: 30px;";
html += "}";
html += "#lightSensor {";
html += "  position: absolute;";
html += "  background-color:rgba(255,255,0,0.5);";
html += "  margin-top: 660px;";
html += "  margin-left: 425px;";
html += "  font-size: 50px;";
html += "}";
html += "#lightSensor1 {";
html += "  position: absolute;";
```

```
html += "  background-color:rgba(255,255,0,0.5);";
html += "  margin-top: 860px;";
html += "  margin-left: 135px;";
html += "  font-size: 50px;";
html += "}";
html += "#lightSensor2 {";
html += "  position: absolute;";
html += "  background-color:rgba(255,255,0,0.5);";
html += "  margin-top: 860px;";
html += "  margin-left: 725px;";
html += "  font-size: 50px;";
html += "}";
html += "#lightSensor3 {";
html += "  position: absolute;";
html += "  background-color:rgba(255,255,0,0.5);";
html += "  margin-top: 1120px;";
html += "  margin-left: 425px;";
html += "  font-size: 50px;";
html += "}";
html += "#north {";
html += "  position: absolute;";
html += "  margin-top: 1250px;";
html += "  margin-left: 420px;";
html += "}";
html += "#east {";
html += "  position: absolute;";
html += "  margin-top: 1445px;";
html += "  margin-left: 610px;";
html += "}";
html += "#south {";
html += "  position: absolute;";
html += "  margin-top: 1635px;";
html += "  margin-left: 420px;";
```

```
 html += "}";
 html += "#west {";
 html += "  position: absolute;";
 html += "  margin-top: 1445px;";
 html += "  margin-left: 240px;";
 html += "}";
 html += "button {";
 html += "  font-size: 50px;";
 html += "  width: 100px;";
 html += "  height: 100px;";
 html += "}";
 html += "</style>";
 html += "<div id='backgroundBlock'>";
 html += "<div id='dataContainer' style='display:flex; justify-content: space-between;
align-items: flex-start;'>";
 html += "<div style='display: flex; flex-direction: column; align-items: flex-start;'>";
 html += "<p id='busVoltage'></p>";
 html += "<p id='current'></p>";
 html += "<p id='busVoltage1'></p>";
 html += "<p id='current1'></p>";
 html += "<p id='lightSensor'></p>";
 html += "<p id='lightSensor1'></p>";
 html += "<p id='lightSensor2'></p>";
 html += "<p id='lightSensor3'></p>";
 html += "</div>";
 html += "</div>";
 html += "<p id='north'><a href=\"/North\"><button>N</button></a></p>";
 html += "<p id='east'><a href=\"/East\"><button>E</button></a></p>";
 html += "<p id='south'><a href=\"/South\"><button>S</button></a></p>";
 html += "<p id='west'><a href=\"/West\"><button>W</button></a></p>";
 html += "</div>";
 html += "</body></html>";
 server.send(200, "text/html", html);
```

```
}

void handleN() {
  Serial.write('7');
  bot.sendMessage(CHAT_ID, "Вгору");
}

void handleE() {
  Serial.write('5');
  bot.sendMessage(CHAT_ID, "Праворуч");
}
void handleS() {
  Serial.write('8');
  bot.sendMessage(CHAT_ID, "Вниз");
}

void handleW() {
  Serial.write('4');
  bot.sendMessage(CHAT_ID, "Ліворуч");
}

void handleDataRequest() {
  String response = String(busVoltage[1]) + ";" +
                    String(current[1]) + ";" +
                    String(busVoltage[2]) + ";" +
                    String(current[2]) + ";" +
                    String(lightSensor) + ";" +
                    String(lightSensor1) + ";" +
                    String(lightSensor2) + ";" +
                    String(lightSensor3);
  server.send(200, "text/plain", response);
}
```

```
void Server() {
  Serial.begin(115200);

  if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin(_ssid.c_str(), _password.c_str());
    while (WiFi.waitForConnectResult() != WL_CONNECTED) {
      delay(1000);
      Serial.println("Connecting to WiFi...");
    }
    Serial.println(WiFi.localIP());
  }

  server.on("/", handleRoot);
  server.on("/North", handleN);
  server.on("/East", handleE);
  server.on("/South", handleS);
  server.on("/West", handleW);
  server.on("/data", handleDataRequest);
  server.begin();
}

void setup() {
  Serial.begin(115200);
  Serial.println("");
  Serial.println("Start 1-WIFI");
  WIFIinit();
  Server();
  configTime(0, 0, "pool.ntp.org");
  client.setTrustAnchors(&cert);
  bot.sendMessage(CHAT_ID, "Local IP: http://IP");
}

void handleNewMessages(int numNewMessages) {
```

```
 for (int i = 0; i < numNewMessages; i++) {
   String chat_id = String(bot.messages[i].chat_id);
   String text = bot.messages[i].text;
   if (text.equals("Status")) {

     String message = "Bus Voltage 1: " + String(busVoltage[1]) + " V\n";
     message += "Current 1: " + String(current[1]) + " mA\n";
     message += "Bus Voltage 2: " + String(busVoltage[2]) + " V\n";
     message += "Current 2: " + String(current[2]) + " mA\n";
     message += "North: " + String(lightSensor) + "\n";
     message += "West: " + String(lightSensor1) + "\n";
     message += "East: " + String(lightSensor2) + "\n";
     message += "South: " + String(lightSensor3) + "\n";
     bot.sendMessage(chat_id, message);
   } else if (text.equals("N")) {
     Serial.write('7');
     bot.sendMessage(CHAT_ID, "Вгору");
   } else if (text.equals("E")) {
     Serial.write('5');
     bot.sendMessage(CHAT_ID, "Праворуч");
   } else if (text.equals("S")) {
     Serial.write('8');
     bot.sendMessage(CHAT_ID, "Вниз");
   } else if (text.equals("W")) {
     Serial.write('4');
     bot.sendMessage(CHAT_ID, "Ліворуч");
   }
 }
}
void loop() {
 server.handleClient();
 while (Serial.available() > 0) {
   char dataType = Serial.read();
```

```
  if (dataType == 'V') {
    char sensorIndex = Serial.read();
    busVoltage[sensorIndex - '0'] = Serial.parseFloat();
  } else if (dataType == 'A') {
    char sensorIndex = Serial.read();
    current[sensorIndex - '0'] = Serial.parseFloat();
  } else if (dataType == 'N') {
    lightSensor = Serial.parseInt();
  } else if (dataType == 'S') {
    lightSensor1 = Serial.parseInt();
  } else if (dataType == 'E') {
    lightSensor2 = Serial.parseInt();
  } else if (dataType == 'W') {
    lightSensor3 = Serial.parseInt();
  }
 }
 if (millis() > bot_lasttime + BOT_MTBS)  {
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  if (numNewMessages > 0) {
    handleNewMessages(numNewMessages);
  }
  bot_lasttime = millis();
 }

 if (Serial.available() > 0) {
  String data = Serial.readStringUntil('\n');
  if (!data.isEmpty()) {
    server.send(200, "text/plain", data);
  }
 }
 ArduinoOTA.handle();
}
```

Додаток Б

ATmega328P:

```
#include <Wire.h>
#include <Adafruit_INA219.h>

#define INA219_ADDRESS_1 0x40
#define INA219_ADDRESS_2 0x44

Adafruit_INA219 ina1(INA219_ADDRESS_1);
Adafruit_INA219 ina2(INA219_ADDRESS_2);

int lightSensor = 0;
int lightSensor1 = 0;
int lightSensor2 = 0;
int lightSensor3 = 0;
int lightsensors;
int printCounter = 0;

unsigned long startTime = 0;
const unsigned long activeTime = 3000;

void setup() {
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(115200);
  Wire.begin();
  ina1.begin();
  ina2.begin();
}
```

```
void loop() {
  if (Serial.available() > 0) {
    char inChar = Serial.read();
    if (inChar == '7') {
  digitalWrite(7, LOW);
  startTime = millis();
} else if (inChar == '5') {
  digitalWrite(13, LOW);
  startTime = millis();
} else if (inChar == '8') {
  digitalWrite(8, LOW);
  startTime = millis();
} else if (inChar == '4') {
  digitalWrite(12, LOW);
  startTime = millis();
}
  }

  if (millis() - startTime >= activeTime) {
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);
    digitalWrite(12, HIGH);
    digitalWrite(13, HIGH);
  }

  Serial.print('N');
  Serial.print(analogRead(A0));
  Serial.print('S');
  Serial.print(analogRead(A1));
  Serial.print('E');
  Serial.print(analogRead(A2));
  Serial.print('W');
  Serial.print(analogRead(A3));
```

```
if (Serial.available()> 0) {
  lightsensors = Serial.parseInt();
}

Serial.print('V');
Serial.print('1');
Serial.print(ina1.getBusVoltage_V());
Serial.print('A');
Serial.print('1');
Serial.print(ina1.getCurrent_mA());

Serial.print('V');
Serial.print('2');
Serial.print(ina2.getBusVoltage_V());
Serial.print('A');
Serial.print('2');
Serial.println(ina2.getCurrent_mA());

printCounter++;

if (printCounter >= 10) {
  while (Serial.available()) {
    Serial.read();
  }
  printCounter = 0;
}

delay(500);
Serial.setTimeout(500);
}
```